d cit ab 1-19

1. 5,911,052, Jun. 8, 1999, Split transaction snooping bus protocol; Ashok Singhal, et al., 710/113 [IMAGE AVAILABLE]

US PAT NO: 5,911,052 [IMAGE AVAILABLE] L3: 1 of 19

ABSTRACT:
A split **transaction** snooping bus protocol and architecture is provided for use in a system having one or many such buses. Circuit boards including CPU or other devices and/or distributed memory, data input/output buffers, **queues** including request tag **queues**, coherent input **queues** ("CIQ"), and address controller implementing address bus **arbitration** plug-into one or more split **transaction** snooping bus systems. All devices snoop on the address bus to learn whether an identified line is owned or shared, and an appropriate owned/shared signal is issued. Receipt of an ignore signal blocks CIQ loading of a **transaction** until the **transaction** is reloaded and ignore is deasserted. Ownership of a requested memory line transfers immediately at time of request. Asserted requests are **queued** such that state **transactions** on the address bus occur atomically logically without dependence upon the request. Subsequent requests for the same data are tagged to become the responsibility of the owner-requestor. A subsequent requestor's activities are not halted awaiting grant and completion of an earlier request **transaction**. Processor-level cache changes state upon receipt of **transaction** data. A single multiplexed **arbitration** bus carries address bus and data bus request **transactions**, which **transactions** are each two-cycles in length.

2. 5,901,295, May 4, 1999, Address and data bus arbiter for pipelined transactions on a split bus; Farid A. Yazdy, 710/113 [IMAGE AVAILABLE]

US PAT NO: 5,901,295 [IMAGE AVAILABLE] L3: 2 of 19

ABSTRACT:
An **arbiter** employs both an address bus **arbiter** and a data bus **arbiter** for supporting pipelined, split bus **transactions**. The address **arbiter** may be implemented using a state machine. A first through third states of the state machine grant the address bus to a respective first through third bus masters, each having a different priority associated therewith. Idle states are interposed between states. The data bus **arbiter** may be implemented using a circular **FIFO** having a plurality of pointers to keep track of present and future bus masters using the data bus.

3. 5,878,237, Mar. 2, 1999, Apparatus, method and system for a comuter CPU and memory to PCI bridge having a pluarlity of physical PCI buses; Sompong P. Olarig, 710/128, 126, 129 [IMAGE AVAILABLE]

US PAT NO: 5,878,237 [IMAGE AVAILABLE] L3: 3 of 19

ABSTRACT:
A core logic chip set in a computer system provides a bridge between processor host and memory buses and a plurality of peripheral component interconnect ("PCI") buses capable of operating at 66 MHz. Each of the plurality of PCI buses have the same logical bus number. The core logic chip set has an **arbiter** having Request ("REQ") and Grant ("GNT") signal lines for each PCI device connected to the plurality of PCI

physical buses. Each of the plurality of PCI buses has its own read and write **queues** to provide **transaction** concurrency of PCI devices on different ones of the plurality of PCI buses when the **transaction** addresses are not the same or are M byte aligned. Upper and lower memory address range registers store upper and lower memory addresses associated with each PCI device. Whenever a **transaction** occurs, the **transaction** address is compared with the stored range of memory addresses. If a match between addresses is found then strong ordering is used. If no match is found then weak ordering may be used to improve **transaction** latency times. PCI device to PCI device **transactions** may occur without being starved by CPU host bus to PCI bus **transactions**.

4. 5,835,739, Nov. 10, 1998, Method and apparatus for maintaining transaction ordering and arbitrating in a bus bridge; D. Michael Bell, et al., 710/128, 129 [IMAGE AVAILABLE]

US PAT NO: 5,835,739 [IMAGE AVAILABLE] L3: 4 of 19

ABSTRACT:
A bus bridge situated between two buses includes two **queues**: an outbound request **queue** and an inbound request **queue**. Requests originating on the first bus which target a destination on the second bus are placed into the outbound request **queue**. Requests originating on the second bus which target a destination on the first bus are placed into the inbound request **queue**. A **transaction arbitration** unit (TAU) within the bridge maintains **transaction** ordering and avoids deadlocks. The TAU determines whether requests can be placed in the inbound request **queue**. The TAU also determines whether requests originating on the first bus can be responded to immediately or whether the agent originating the request must wait for a reply. In addition, the TAU includes logic for determining whether a request in the outbound request **queue** can be executed on the second bus. The TAU determines whether posting to the inbound request **queue** is enabled or disabled; whether any posted **transactions** exist in the inbound request **queue**; and whether ownership of the second bus is available.

5. 5,787,237, Jul. 28, 1998, Uniform interface for conducting communications in a heterogeneous computing network; Paul E. Reilly, 395/112, 114; 710/11, 39, 65 [IMAGE AVAILABLE]

US PAT NO: 5,787,237 [IMAGE AVAILABLE] L3: 5 of 19

ABSTRACT:
A network printing system for enhancing two-way communication between host computers and printers connected to a network. More particularly, the network printing system provides an imaging device protocol (DP) which enables various network service protocol/ports for host computers to communicate with the printer by "seamless plug and play" connectivity. IDP operates independently of the network layers below and only requires that the transport protocol/port be bidirectional. IDP allows all of the incoming print **job** information to be placed in the print **queue** regardless of the protocol for a wide variety of heterogenous network protocols. As a result, the network printing system enables print **jobs** from host computers connected to the network by a wide variety of heterogenous network protocols to be fairly **arbitrated** at the printer.

6. 5,724,587, Mar. 3, 1998, System for controlling task execution in a host processor based upon the maximum DMA resources available to a digital signal processor; Donald Edward Carmon, et al., 709/104, 100 [IMAGE AVAILABLE]

US PAT NO: 5,724,587 [IMAGE AVAILABLE] L3: 6 of 19

ABSTRACT:
A multi-media user **task** (host) computer is interfaced to a high speed DSP which provides support functions to the host computer via an interprocessor DMA bus master and controller. Support of multiple dynamic hard real-time signal processing **task** requirements are met by posting signal processor support **task** requests from the host processor through the interprocessor DMA controller to the signal processor and its operating system. The signal processor builds data transfer packet request execution lists in a partitioned **queue** in its own memory and executes internal signal processor **tasks** invoked by users at the host system by extracting signal sample data from incoming data packets presented by the interprocessor DMA controller in response to its execution of the DMA packet transfer request **queues** built by the signal processor in the partitioned **queue**. Processed signal values etc. are extracted from signal processor memory by the DMA interprocessor controller executing the partitioned packet request lists and delivered to the host processor. A very large number of packet transfers in support of numerous user **tasks** and implementing a very large number of DMA channels is thus made possible while avoiding the need for **arbitration** between the channels on the part of the signal processor or the host processor.

7. 5,724,583, Mar. 3, 1998, System for handling requests for DMA data transfers between a host processor and a digital signal processor; Donald Edward Carmon, et al., 709/100; 710/25, 28 [IMAGE AVAILABLE]

US PAT NO: 5,724,583 [IMAGE AVAILABLE] L3: 7 of 19

ABSTRACT:
A multi-media user **task** (host) computer is interfaced to a high speed DSP which provides support functions to the host computer via an interprocessor DMA bus master and controller. Support of multiple dynamic hard real-time signal processing **task** requirements are met by posting signal processor support **task** requests from the host processor through the interprocessor DMA controller to the signal processor and its operating system. The signal processor builds data transfer packet request execution lists in a partitioned **queue** in its own memory and executes internal signal processor **tasks** invoked by users at the host system by extracting signal sample data from incoming data packets presented by the interprocessor DMA controller in response to its execution of the DMA packet transfer request **queues** built by the signal processor in the partitioned **queue**. Processed signal values etc. are extracted from signal processor memory by the DMA interprocessor controller executing the partitioned packet request lists and delivered to the host processor. A very large number of packet transfers in support of numerous user **tasks** and implementing a very large number of DMA channels is thus made possible while avoiding the need for **arbitration** between the channels on the part of the signal processor or the host processor.

8. 5,708,783, Jan. 13, 1998, Data bus arbiter for pipelined transactions on a split bus; Farid A. Yazdy, 710/113 [IMAGE AVAILABLE]

US PAT NO: 5,708,783 [IMAGE AVAILABLE] L3: 8 of 19

ABSTRACT:
A data bus **arbiter** for supporting pipelined **transactions** employs a circular **FIFO** for storing bus requests. The **arbiter** includes two pointers which reference the entries of the **FIFO**. A first pointer is incremented upon detection of the end of a bus cycle. A second pointer is incremented when a new bus cycle is started.

9. 5,697,040, Dec. 9, 1997, Print job intermixing within marking machine; Douglas T. Rabjohns, et al., 399/382, 82 [IMAGE AVAILABLE]

US PAT NO: 5,697,0[redacted] [IMAGE AVAILABLE] [redacted] 9 of 19

ABSTRACT:
A method of **arbitrarily** interleaving the images of a second **job** with the images of a first **job** that is being completed at a first output station. A **queue** of **jobs** submitted to a marking machine for completion is scanned to determine if any of the **queue** of **jobs** can be interleaved with the first **job** during the first **job** production run. If so, one or more of the **jobs** in the **queue** are scheduled to be interleaved during the first **job** production run for completion at a second output station.

10. 5,574,868, Nov. 12, 1996, Bus grant prediction technique for a split transaction bus in a multiprocessor computer system; Suresh Marisetty, 710/118, 107, 117; 711/167, 169 [IMAGE AVAILABLE]

US PAT NO: 5,574,868 [IMAGE AVAILABLE] L3: 10 of 19

ABSTRACT:
A early bus grant prediction technique combines the operating advantages of both a split **transaction** bus and a simple shared bus. When a read request is generated by a memory access requester, an early bus request is generated for the impending data transfer. The early bus request is provided to bus grant prediction and **arbitration** logic that determines whether or not the bus will be available at the time the requested data has been retrieved and is ready for transfer. If the bus is available, the retrieved data is routed immediately to the memory bus for a fly-by transfer. On the other hand, if the bus is not available, the data is routed to a **FIFO** buffer to be transferred when the bus is available.

11. 5,546,546, Aug. 13, 1996, Method and apparatus for maintaining transaction ordering and arbitrating in a bus bridge; D. Michael Bell, et al., 710/112, 100, 105 [IMAGE AVAILABLE]

US PAT NO: 5,546,546 [IMAGE AVAILABLE] L3: 11 of 19

ABSTRACT:
A bus bridge situated between two buses includes two **queues**: an outbound request **queue** and an inbound request **queue**. Requests originating on the first bus which target a destination on the second bus are placed into the outbound request **queue**. Requests originating on the second bus which target a destination on the first bus are placed into the inbound request **queue**. A **transaction arbitration** unit (TAU) within the bridge maintains **transaction** ordering and avoids deadlocks. The TAU determines whether requests can be placed in the inbound request **queue**. The TAU also determines whether requests originating on the first bus can be responded to immediately or whether the agent originating the request must wait for a reply. In addition, the TAU includes logic for determining whether a request in the outbound request **queue** can be executed on the second bus. The TAU determines whether posting to the inbound request **queue** is enabled or disabled; whether any posted **transactions** exist in the inbound request **queue**; and whether ownership of the second bus is available.

12. 5,485,586, Jan. 16, 1996, Queue based arbitration using a FIFO data structure; David L. A. Brash, et al., 710/112; 340/825.5; 364/240.1, 242.6, 242.92, DIG.1; 370/462; 710/113, 240 [IMAGE AVAILABLE]

US PAT NO: 5,485,586 [IMAGE AVAILABLE] L3: 12 of 19

ABSTRACT:
A **queue** based **arbiter** to **arbitrate** between N devices of a computer system for access to a system bus which eliminates the need to maintain a history of bus **transactions** by queuing bus requests to

track when a bus reque[redacted] is posted. The **arbiter** provide[redacted] fair access to the bus by maintain[redacted] a **queue** of requests that come[redacted] from each resource in the computer system. This is accomplished by continually sampling the individual request lines of the devices to determine if a device is requesting access to the bus. Each time the **arbiter** detects a request from a device it puts an entry representative of the specific device that has requested the bus into a **queue** that has at least N entries. Requests are granted in the order that they are **queued**.

13. 5,404,522, Apr. 4, 1995, System for constructing a partitioned queue of DMA data transfer requests for movements of data between a host processor and a digital signal processor; Donald E. Carmon, et al., 709/107; 364/228.6, 242.3, 244.3, 246.3, 271.6, DIG.1; 710/22 [IMAGE AVAILABLE]

US PAT NO: 5,404,522 [IMAGE AVAILABLE] L3: 13 of 19

ABSTRACT:
A multi-media user **task** (host) computer is interfaced to a high speed DSP which provides support functions to the host computer via an interprocessor DMA bus master and controller. Support of multiple dynamic hard real-time signal processing **task** requirements are met by posting signal processor support **task** requests from the host processor through the interprocessor DMA controller to the signal processor and its operating system. The signal processor builds data transfer packet request execution lists in a partitioned **queue** in its own memory and executes internal signal processor **tasks** invoked by users at the host system by extracting signal sample data from incoming data packets presented by the interprocessor DMA controller in response to its execution of the DMA packet transfer request **queues** built by the signal processor in the partitioned **queue**. Processed signal values etc. are extracted from signal processor memory by the DMA interprocessor controller executing the partitioned packet request lists and delivered to the host processor. A very large number of packet transfers in support of numerous user **tasks** and implementing a very large number of DMA channels is thus made possible while avoiding the need for **arbitration** between the channels on the part of the signal processor or the host processor.

14. 5,333,296, Jul. 26, 1994, Combined queue for invalidates and return data in multiprocessor system; Gregg Bouchard, et al., 711/171; 364/239.8, 243.41, 247.7, 254.5, DIG.1; 711/117 [IMAGE AVAILABLE]

US PAT NO: 5,333,296 [IMAGE AVAILABLE] L3: 14 of 19

ABSTRACT:
A pipelined CPU executing instructions of variable length, and referencing memory using various data widths. Macroninstruction pipelining is employed (instead of microinstruction pipelining), with queuing between units of the CPU to allow flexibility in instruction execution times. A wide bandwidth is available for memory access; fetching 64-bit data blocks on each cycle. A hierarchical cache arrangement is used, increasing the likelihood of a cache hit. A writeback cache is used (instead of writethrough) and writeback is allowed to proceed even though other accesses are suppressed due to **queues** being full. Separate **queues** are provided for the return data from memory and cache invalidates, yet the order or bus **transactions** is maintained by a pointer arrangement. The bus protocol used by the CPU to communicate with the system bus is of the pended type, with **transactions** on the bus identified by an ID field specifying the originator, and **arbitration** for bus grant goes one simultaneously with address/data **transactions** on the bus.

15. 5,317,720, May 31, 1994, Processor system with writeback cache using writeback and non writeback transactions stored in separate queues;

Rebecca L. Stamm, et al., 711/143; 364/243.41, 243.45, [redacted].5, DIG.1; 711/122 [IMAGE AVAILAB[redacted]

US PAT NO: 5,317,720 [IMAGE AVAILABLE] L3: 15 of 19

ABSTRACT:
A pipelined CPU executing instructions of variable length, and referencing memory using various data widths. A writeback cache is used (instead of writethrough) in a hierarchical cache arrangement, and writeback is allowed to proceed even though other accesses are suppressed due to **queues** being full. Separate **queues** are provided for the return data from memory and cache invalidates, yet the order or bus **transactions** is maintained by a pointer arrangement. The bus protocol used by the CPU to communicate with the system bus is of the pended type, with **transactions** on the bus identified by an ID field specifying the originator, and **arbitration** for bus grant goes one simultaneously with address/data **transactions** on the bus.

16. 5,287,477, Feb. 15, 1994, Memory-resource-driven arbitration; Leith L. Johnson, et al., 711/157; 364/240.8, 242.1, 243, DIG.1 [IMAGE AVAILABLE]

US PAT NO: 5,287,477 [IMAGE AVAILABLE] L3: 16 of 19

ABSTRACT:
A method and apparatus to improve memory performance in a computer bus system. Memory is divided into interleaved blocks and memory addresses are mapped into block identification numbers. Master devices keep track of which parts of memory are busy by storing memory block identification numbers in local **queues** whenever memory is accessed. Block identification numbers are removed from local **queues** when the memory **transaction** is complete. Master devices **arbitrate** for access to the bus for memory **transactions** only if the target memory block identification number is not in the local **queue**.

17. 5,155,843, Oct. 13, 1992, Error transition mode for multi-processor system; Rebecca L. Stamm, et al., 714/5; 364/228, 231.8, 232.8, 239, 239.8, 239.9, 240, 240.2, 240.8, 241.9, 242.6, 242.92, 243, 243.41, 243.42, 243.44, 244, 244.3, 244.6, 244.9, 247, 247.8, 251, 251.3, 254.9, 259, 259.2, 259.5, 259.7, 259.9, 261.3, 261.4, 261.5, 261.6, 262.4, 262.7, 262.8, 262.81, 263.1, 265, 265.3, 268, 268.3, 268.5, 271.5, 280.8, 927.8, 931.4, 931.49, 931.55, 934, 934.2, 943.9, 944.92, 948, 948.34, 957, 957.6, 964, 964.2, 964.22, 964.32, 964.34, 964.341, 964.342, 964.343, DIG.1, DIG.2 [IMAGE AVAILABLE]

US PAT NO: 5,155,843 [IMAGE AVAILABLE] L3: 17 of 19

ABSTRACT:
A pipelined CPU executing instructions of variable length, and referencing memory using various data widths. Macroinstruction pipelining is employed (instead of microinstruction pipelining), with queueing between units of the CPU to allow flexibility in instruction execution times. A wide bandwidth is available for memory access; fetching 64-bit data blocks on each cycle. A hierarchical cache arrangement has an improved method of cache set selection, increasing the likelihood of a cache hit. A writeback cache is used (instead of writethrough) and writeback is allowed to proceed even though other accesses are suppressed due to **queues** being full. A branch prediction method employs a branch history table which records the taken vs. not-taken history of branch opcodes recently used, and uses an empirical algorithm to predict which way the next occurrence of this branch will go, based upon the history table. A floating point processor function is integrated on-chip, with enhanced speed due to a bypass technique; a trial mini-rounding is done on low-order bits of the result, and if correct, the last stage of the floating point processor can be bypassed, saving one cycle of latency.

For CAL type instructions, a method for determining which registers need to be saved is executed in a minimum number of cycles, examining groups of register mask bits at one time. Internal processor registers are accessed with short (byte width) addresses instead of full physical addresses as used for memory and I/O references, but off-chip processor registers are memory-mapped and accessed by the same busses using the same controls as the memory and I/O. In a non-recoverable error detected by ECC circuits in the cache, an error transition mode is entered wherein the cache operates under limited access rules, allowing a maximum of access by the system for data blocks owned by the cache, but yet minimizing changes to the cache data so that diagnostics may be run. Separate **queues** are provided for the return data from memory and cache invalidates, yet the order or bus **transactions** is maintained by a pointer arrangement. The bus protocol used by the CPU to communicate with the system bus is of the pended type, with **transactions** on the bus identified by an ID field specifying the originator, and **arbitration** for bus grant goes one simultaneously with address/data **transactions** on the bus.

18. 5,006,982, Apr. 9, 1991, Method of increasing the bandwidth of a packet bus by reordering reply packets; Ronald J. Ebersole, et al., 710/263; 364/229.2, 230.1, 242.8, 242.92, 284.1, 284.3, DIG.1 [IMAGE AVAILABLE]

US PAT NO: 5,006,982 [IMAGE AVAILABLE] L3: 18 of 19

ABSTRACT:
A data processor bus in which information is transferred between agents attached to the bus by issuing request packets that request data from an agent on the bus and reply packets that return data requested by a request packet. A control method mixes request-and-reply packets on the bus by determining the use of a next-bus cycle using **arbitration**, reply deferral, and specification lines and the state of a grant **queue** and a pipe **queue** in accordance with a specified protocol. A request is forced to take the next available bus cycle upon the condition that there is an agent identified in the great **queue** and the pipeline **queue** is not full. A reply packet is forced to take the next available bus cycle upon the condition that the pipeline **queue** is full. A reply packet is forced to take the next available bus cycle upon the condition that the grant **queue** is empty and the pipeline **queue** is not empty. Giving requests precedence over replies to allows the pipeline to be kept as full as possible. A replying agent assigned to the highest priority slot 1 in the pipeline **queue** is allowed to defer its own slot in the pipeline **queue** until a later time to thereby permit a **transaction** in Slot 2 of the pipeline **queue** to be completed before the one ahead of it.

19. 4,884,228, Nov. 28, 1989, Flexible instrument control system; James C. Stanley, et al., 702/123; 364/221, 221.7, 230.3, 232.7, 232.8, 232.9, 237.2, 242.2, 242.6, 242.7, 242.91, 245.6, 261, 280, 281.3, 281.7, DIG.1 [IMAGE AVAILABLE]

US PAT NO: 4,884,228 [IMAGE AVAILABLE] L3: 19 of 19

ABSTRACT:
Operation of a microcomputer-based instrument is controlled by software including a control interface system, a command execution system, and a steady state system. The steady state system includes a group of subsystems operating as concurrent **tasks**, each carrying out various instrument operations. The control interface system includes a set of subsystems, each responsive to configuration control input signals from a separate control signal source, and each converting configuration control signals into "configuration commands". Each command identifies a particular procedure for configuring the operating state of one or more instrument hardware or software steady state subsystems of the

instrument. Each command is stored in memory in one of several **queues** (one for each control interface subsystem) until the procedure it invokes is carried out. The command execution system includes an **arbitrator** routine which selects commands stored in the **queues** according to a predetermined priority scheme and invokes command execution subroutines for carrying out procedures identified by the selected commands. When a command execution subroutine is invoked, it reconfigures one or more instrument hardware or software subsystems according to configuration parameters included in the command.

d 1-6 cit ab

1. 5,754,803, May 19, 1998, Parallel packetized intermodule arbitrated high speed control and data bus; Robert T. Regis, 710/119, **107** [IMAGE AVAILABLE]

US PAT NO: 5,754,803 [IMAGE AVAILABLE] L6: 1 of 6

ABSTRACT:
A parallel packetized intermodule **arbitrated** high speed control data bus system which allows high speed communications between microprocessor modules in a more complex digital processing environment. The system features a simplified hardware architecture featuring fast **FIFO** queuing operating at 12.5 MHz, TTL CMOS compatible level clocking signals, single bus **master arbitration**, synchronous clocking, DMA, and unique module addressing for multiprocessor systems. The system includes a parallel data bus with sharing bus **masters** residing on each processing module decreeing the communication and data transfer protocols. Bus **arbitration** is performed over a dedicated serial **arbitration** line and each requesting module competes for access to the parallel data bus by placing the address of the requesting module on the **arbitration** line and monitoring the **arbitration** line for collisions.

2. 5,724,587, Mar. 3, 1998, System for controlling task execution in a host processor based upon the maximum DMA resources available to a digital signal processor; Donald Edward Carmon, et al., 709/104, **100** [IMAGE AVAILABLE]

US PAT NO: 5,724,587 [IMAGE AVAILABLE] L6: 2 of 6

ABSTRACT:
A multi-media user task (host) computer is interfaced to a high speed DSP which provides support functions to the host computer via an interprocessor DMA bus **master** and controller. Support of multiple dynamic hard real-time signal processing task requirements are met by posting signal processor support task requests from the host processor through the interprocessor DMA controller to the signal processor and its operating system. The signal processor builds data transfer packet request execution lists in a partitioned **queue** in its own memory and executes internal signal processor tasks invoked by users at the host system by extracting signal sample data from incoming data packets presented by the interprocessor DMA controller in response to its execution of the DMA packet transfer request **queues** built by the signal processor in the partitioned **queue**. Processed signal values etc. are extracted from signal processor memory by the DMA interprocessor controller executing the partitioned packet request lists and delivered to the host processor. A very large number of packet transfers in support of numerous user tasks and implementing a very large number of DMA channels is thus made possible while avoiding the need for **arbitration** between the channels on the part of the signal processor or the host processor.

3. 5,724,583, Mar. 3, 1998, System for handling requests for DMA data transfers between a host processor and a digital signal processor; Donald Edward Carmon, et al., **709/100**; 710/25, 28 [IMAGE AVAILABLE]

US PAT NO: 5,724,583 [IMAGE AVAILABLE] L6: 3 of 6

ABSTRACT:
A multi-media user task (host) computer is interfaced to a high speed DSP which provides support functions to the host computer via an interprocessor DMA bus **master** and controller. Support of multiple dynamic hard real-time signal processing task requirements are met by posting signal processor support task requests from the host processor through the interprocessor DMA controller to the signal processor and its operating system. The signal processor builds data transfer packet request execution lists in a partitioned **queue** in its own memory and executes internal signal processor tasks invoked by users at the host system by extracting signal sample data from incoming data packets presented by the interprocessor DMA controller in response to its execution of the DMA packet transfer request **queues** built by the signal processor in the partitioned **queue**. Processed signal values etc. are extracted from signal processor memory by the DMA interprocessor controller executing the partitioned packet request lists and delivered to the host processor. A very large number of packet transfers in support of numerous user tasks and implementing a very large number of DMA channels is thus made possible while avoiding the need for **arbitration** between the channels on the part of the signal processor or the host processor.

4. 5,564,025, Oct. 8, 1996, Apparatus for arbitrating requests for access from slave units by associating the requests with master units and determining the relative pendency thereof in a radio base station transceiver; Karsten De Freese, et al., **710/110**; 340/825.06; 364/230.4, 242.8, 242.92, DIG.1; 710/40, 123 [IMAGE AVAILABLE]

US PAT NO: 5,564,025 [IMAGE AVAILABLE] L6: 4 of 6

ABSTRACT:
The radio base station (BSI) in a cellular radio communication system such as a GSM system must perform considerable real time processing, e.g. for channel coding and decoding, which requires a considerable number of processors (SC01, SC02, OC0). For optimal deployment of the processors and to facilitate extensions and modifications of the system, the radio base station (BS1) has an internal **arbitration** bus (IPB) for coupling **master** and **slave** control units and signal processing units to the processors and to channel coders (CHC1, CHC2, CHC3, CHC4) or other resources. A **master** and **slave arbitration** system is thereby achieved wherein requests from **master** and **slave** units can be **queued** flexibly, and in which optimal use is made of available resources. By using a RAM table for assigning **master** and **slave** units to each other, further flexibility in use of resources is achieved. Adaptive polling of requesting units may be employed in order to increase system throughput.

5. 5,475,850, Dec. 12, 1995, Multistate microprocessor bus arbitration signals; Mitchell A. Kahn, **710/110**; 364/240.5, 242.6, 242.92, 243.4, DIG.1; 710/113, 241 [IMAGE AVAILABLE]

US PAT NO: 5,475,850 [IMAGE AVAILABLE] L6: 5 of 6

ABSTRACT:
A microprocessor bus **arbitration** communications scheme for enhancing efficiency and performance of a multi-**master** bus system, typically within a computer system, including a central processing unit ("CPU") being a primary bus **master**, a bus **arbiter** and at least one alternative bus **master** coupled together by a bus. The CPU includes an internal memory element, a bus **queue** and bus control logic which collectively operate to generate a plurality of microprocessor bus **arbitration** signals to the bus **arbiter**. These microprocessor bus **arbitration** signals include a first bus **arbitration** signal indicating whether the CPU requires access to the bus and a second bus

arbitration signal indicating that the CPU requires immediate access to the bus.

6. 5,299,315, Mar. 29, 1994, Personal computer with programmable threshold FIFO registers for data transfer; Arthur L. Chin, et al., **710/110** [IMAGE AVAILABLE]

US PAT NO: 5,299,315 [IMAGE AVAILABLE] L6: 6 of 6

ABSTRACT:
This invention relates to personal computers, and more particularly to a personal computer using a **FIFO** registers for data transfer as illustrated by a bus **master** device in the form of a small computer systems interface (SCSI) controller for controlling data transfer with storage memory devices such as fixed or removable media electromagnetic storage devices. In the practice of this invention, the efficiency of a system having a plurality of bus **master** devices is enhanced by providing for a programmable threshold fill condition for a **FIFO** register before **arbitration** for bus control occurs. Thus the invention provides an approach to maximizing the efficiency of data transfer where **FIFO** registers are used.